Exploration vs Exploitation in Bayesian Optimization

Ali Jalali
University of Texas at Austin
alij@mail.utexas.edu

Javad Azimi and Xiaoli Fern
Oregon State University
azimi,xfern@eecs.oregon.edu

March 1, 2013

Abstract

The problem of optimizing unknown costly-to-evaluate functions has been studied for a long time in the context of Bayesian Optimization. Algorithms in this field aim to find the optimizer of the function by asking only a few function evaluations at locations carefully selected based on a posterior model. In this paper, we assume the unknown function is Lipschitz continuous. Leveraging the Lipschitz property, we propose an algorithm with a distinct exploration phase followed by an exploitation phase. The exploration phase aims to select samples that shrink the search space as much as possible. The exploitation phase then focuses on the reduced search space and selects samples closest to the optimizer. Considering the Expected Improvement (EI) as a baseline, we empirically show that the proposed algorithm significantly outperforms EI.

1 Introduction

In many applications such as nanotechnology and finance, etc, we want to optimize an unknown function $f(\cdot)$ that is costly to evaluate over a compact input space. Normal optimization methods cannot be applied to this type of problems since they either need to evaluate the function frequently or they require to know the gradient of the function at each point. In contrast, Bayesian optimization [9, 4] algorithms try to solve this problem with a small number of function evaluations.

Bayesian optimization algorithms have two key components: 1) A posterior model to predict the output value of the function at any arbitrary input point, and 2) A selection criterion to determine which point is going to be evaluated next. The first step of a Bayesian optimization algorithm is to generate a posterior probabilistic model over unobserved points of the function. Gaussian process (GP) [16] has been recently used in most of the literature of Bayesian Optimization as the probabilistic posterior model. In general, GP models the function output for any unobserved point in the input space as a normal random variable, whose mean and variance depend on the location of the point in relation to a set of observed samples. Based on the generated posterior model, a selection criterion is then used to choose the next sample to be evaluated. A number of selection criteria have been proposed in the literature of Bayesian Optimization. They typically work by selecting an example that optimizes some objective function designed to balance between exploring unobserved area and exploiting the promising observed parts of the input space. Maximum probability of improvement [8, 20] and maximum expected improvement (EI) [13] are two successful examples.

In this paper, we focus on the design of the selection criterion for Bayesian Optimization (BO). In particular, we study Bayesian Optimization in a sequential setting [11, 15, 19], where the samples are chosen sequentially and a selection is made only after the function evaluations of the previous samples are revealed. We make a mild assumption that the unknown function is Lipschitz-continuous. Leveraging the Lipschitz property, we design a selection algorithm that operates in two distinct phases: the exploration phase and the exploitation phase.

The exploration phase of the proposed algorithm, at each step, selects a sample that eliminates the largest possible portion of the input space while guaranteing with high probability that the eliminated part does not include the maximizer of the function. Hence, the exploration stage of the algorithm tries to shrink the search space of the function as much as possible. In contrast, the exploitation phase of our algorithm selects the point which is believed to be the closest sample to the optimal point with high probability. Experimental results over 8 real and synthetic benchmarks indicate that the proposed approach is able to outperform the Expected Improvement (EI) criterion, one of the current state-of-the-art BO selection methods. In particular, we show that our algorithm is better than EI both in terms of the mean and variance of the performance.

We also investigate whether combining our exploration stage with EI can boost the performance of EI. However, the results were negative. Sometimes it helps and sometimes it hurts and on average we observe little to no improvement to EI. This is possibly because our exploration method actively aims to eliminate regions from the input space and the EI criterion does not take that into consideration when selecting samples.

The remainder of the paper is organized as follows. In Section 2, we motivate the use of exploration-exploitation Bayesian Optimization by analyzing the behavior of EI. Section 3 introduces our algorithm and provides insights into both theoretical and practical aspects of the algorithm. Experimental evaluation of our algorithm is shown in Section 4. Finally, the paper is concluded in Section 5.

2 Motivating Observation

In this section, we motivate our approach by revealing a key observation about the well known Expected Improvement (EI) algorithm [13]. Using Gaussian Process (GP) [16] as the posterior model of the unknown function, the EI objective is defined as

$$\begin{aligned} EI(x|\mathcal{O}) = (\mu_{x|\mathcal{O}} - y_{\max})\Phi\left(\frac{\mu_{x|\mathcal{O}} - y_{\max}}{\sigma_{x|\mathcal{O}}}\right) \\ + \sigma_{x|\mathcal{O}}\,\phi\left(\frac{\mu_{x|\mathcal{O}} - y_{\max}}{\sigma_{x|\mathcal{O}}}\right), \end{aligned} \tag{1}$$

where, $\mu_{x|\mathcal{O}}$ and $\sigma_{x|\mathcal{O}}$ are the mean and standard deviation associated with the point x by GP, and, $\Phi(\cdot)$ and $\phi(\cdot)$ are standard Gaussian CDF and PDF, respectively. Here, $\mathcal{O} = \{(x_i, f(x_i))\}_{i=1}^n$ is the set of n observed samples $x_\mathcal{O}$ with their function evaluations $f(x_\mathcal{O})$ and define $y_{\max} = \max_{x_i \in x_\mathcal{O}} f(x_i)$. Further, the means and variances are defined as follows:

$$\begin{aligned} \mu_{x|\mathcal{O}} &= k(x, x_\mathcal{O})\, k(x_\mathcal{O}, x_\mathcal{O})^{-1}\, f(x_\mathcal{O}) \\ \sigma^2_{x|\mathcal{O}} &= k(x,x) - k(x, x_\mathcal{O})\, k(x_\mathcal{O}, x_\mathcal{O})^{-1}\, k(x_\mathcal{O}, x), \end{aligned}$$

where $k(\cdot,\cdot)$ is some kernel function. In this paper, we consider Gaussian kernel $k(x_1, x_2) = \exp(-\frac{1}{\ell}\|x_1 - x_2\|_2^2)$.

EI has been widely used and studied; however, there has been always a concern about balancing the exploration and exploitation of EI. The main reason for this concern is that even though the asymptotic convergence of EI is guaranteed under certain conditions [21], if not explored well, EI can be trapped in a local minima when we have finite/limited number of samples. There has been some attempts in the literature to address this concern with varying degrees of success, which we briefly discuss here.

(a) The original of EI is defined as

$$EI(x) = \mathbb{E}\left[(f(x) - y_{\max})\, \mathbb{I}_{\{f(x) - y_{\max} > 0\}}\right],$$

where $\mathbb{I}_{\{.\}}$ is the indicator function. Hence, it measures the expected improvement of the choice of x over the current maximum function evaluations $y_{\max}$ over observed samples. Researchers have proposed to replace $y_{\max}$ with a smaller value to make EI more exploitative and with a larger value to make it more explorative. In particular, Lizotte [12] suggested $y_{\max} + \xi$ and Azimi et al. [2] suggested $(1+\xi)y_{\max}$ to replace $y_{\max}$. However, this approach has not seen much empirical success. Lizotte [12] showed that starting with large values of ξ (to be explorative in the beginning) and cooling it down (to make it more and more exploitative) makes little or no difference in the performance of EI.

(b) On a separate line of work, Schonlau [18] proposed to consider a surrogate function

$$EI_\xi(x) = \mathbb{E}\left[(f(x) - y_{\max})^\xi\, \mathbb{I}_{\{f(x) - y_{\max} > 0\}}\right].$$

For $\xi = 1$, this objective tries to improve over $y_{\max}$ (exploiting mode) and if we decrease ξ it starts to explore uncertain areas (exploration mode). This method is very sensitive to small changes in ξ and except for very specific setup like the one used in [17], there is no systematic way to choose ξ. This makes it nearly impossible to use this method.

(c) The third proposal is to have a "random" exploration phase followed by EI. In this approach, we take a number of random samples before switching to EI. We analyzed this method in Fig. 1. For a fixed budget n_b, we run n_b experiments as follows: first we consider the case where there is 1 random sample followed by $n_b - 1$ samples selected by the EI criterion, next we consider the case where there are 2 random samples followed by $n_b - 2$ EI samples and so on. The purpose of this investigation is to understand whether exploring with random samples prior to selecting with EI can improve the performance of EI, and if so how much exploring is necessary. We

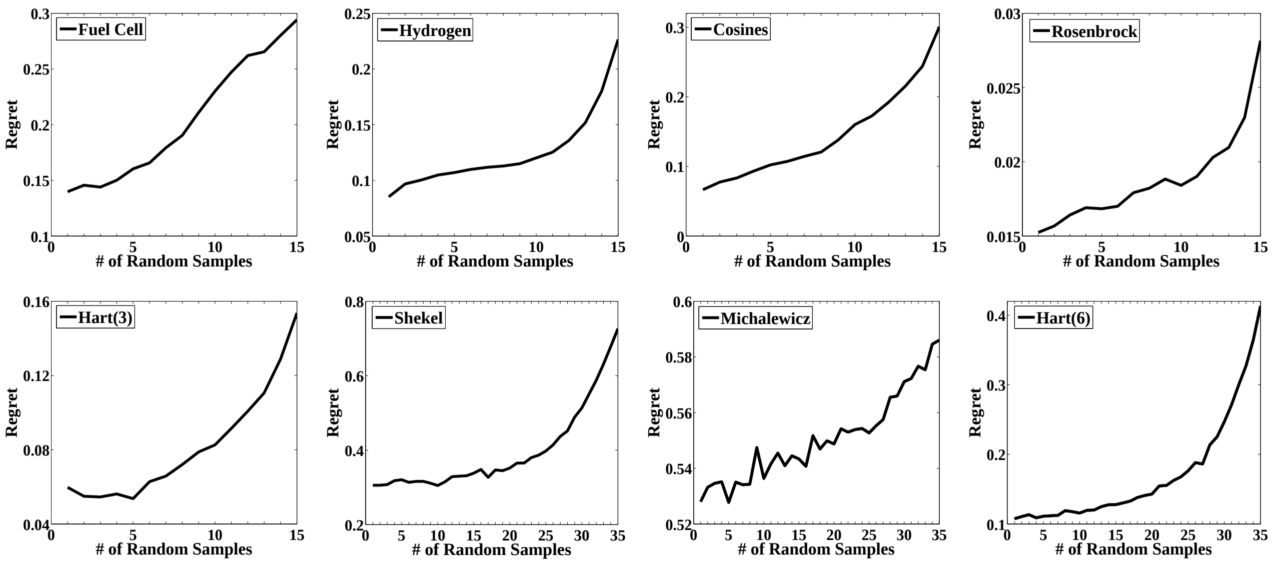

Figure 1: Plot of regret versus the number of random exploration for EI algorithm. For a fixed budget n_b, we run a number of experiments as follows: first we consider the case where there are 1 random samples followed by $n_b - 1$ EI samples, next we consider the case where where there are 2 random samples followed by $n_b - 2$ EI samples and so on. For 2D and 3D functions, we let $n_b = 15$ and for high-dimensional functions, we let $n_b = 35$. This result shows that the best EI performance is when we do not do random exploration.

Algorithm 1 Next Best exploRative Sample (NBRS)

```
Input: Maximum $M$, Lipschitz Constant $L$ and Set of observed samples $\{(x_1, f(x_1)), \ldots, (x_t, f(x_t))\}$
Output: Next best explorative sample $x$

$\mathbb{D}_t = \mathbb{D} - \bigcup_{i=1}^{t} \mathbb{S}(x_i, r_{x_i})$
$x \longleftarrow \operatorname*{argmax}_{x \in \mathbb{D}_t} \textbf{Vol}\left(\mathbb{D}_t \cap \mathbb{S}\left(x, \frac{|M - \mu_{x|\mathcal{O}}| - 1.5\sigma_{x|\mathcal{O}}}{L}\right)\right)$
```

run this experiments on a number of different functions introduced in Section 4. These experiments reveal that "random" exploration never helps EI, since the regret monotonically increases as we increase the number of random samples from 1 to n_b.

Based on the existing literature as well as our empirical investigation of EI discussed above, we would like to know whether or not it is possible to design an algorithm that operates in two naturally defined phases of exploration and exploitation and achieves consistently better performance than EI. We devote the next section to answer this question and introduce our proposed algorithm.

3 Finite Horizon Bayesian Optimization

Not being able to balance the exploration-exploitation, EI might have poor performance especially when the query budget is small. In this section, we propose a two-phase exploration/exploitation algorithm that outperforms EI with its smart exploration and exploitation.

3.1 Exploration

Generally, a good exploration algorithm should be able to shrink the search space, so that we are left with a small region to focus on during the exploit stage. Let $\mathbb{D} = \bigotimes[a_i, b_i] \in \mathbb{R}^d$ be the Cartesian product of intervals $[a_i, b_i]$ for some $a_i < b_i$ and $i \in \{1, 2, \ldots, d\}$. Suppose the unknown function $f : \mathbb{D} \mapsto [m, M]$ (with $f(x^*) = M$) is a Lipschitz function over $\mathbb{D}$ with constant L, that is for all $x_1, x_2 \in \mathbb{D}$, we have

$$|f(x_1) - f(x_2)| \leq L\|x_1 - x_2\|_2.$$

Notice that if the function is not Lipschitz, then there is no hope that we can find the global optimum of $f(\cdot)$ even with infinitely countable evaluations. Thus, the Lipschitz continuity assumption is not a strong assumption. Moreover, functions with larger L are harder to optimize since they change more abruptly over the space.

For any point $x \in \mathbb{D}$, let $r_x = \frac{M-f(x)}{L}$ be the associated radius to the point x. By Lipschitz continuity assumption, we know that $x^* \notin \mathbb{S}(x, r_x)$, where, $\mathbb{S}(x, r_x)$ is the set of all points inside the sphere (or circle) with radius r_x centered at x (and single point x if $r_x \leq 0$); otherwise, the Lipschitz assumption is violated. This means if we have a sample at point x, then we do not need any more samples inside $\mathbb{S}(x, r_x)$.

The expected value of r_x satisfies $\mathbb{E}[r_x] = \frac{|M-\mu_x|}{L}$. Since $f(x)$ is a normal random variable $\mathcal{N}(\mu_x, \sigma_x^2)$, using Hoeffding inequality for all $\epsilon > 0$, we have

$$\mathbb{P}\left[r_x < \frac{|M - \mu_x|}{L} - \epsilon\right] \leq \exp\left(-\frac{2\epsilon^2 L^2}{\sigma_x^2}\right).$$

Replacing ϵ with $1.5\frac{\sigma_x}{L}$, the above inequality entails that with high probability (99%), $r_x \geq \frac{|M-\mu_x|-1.5\sigma_x}{L}$. Hence, a "good" algorithm for exploration should try to find x that maximizes the lower bound on r_x. This choice of x will remove a large volume of points from the search space. Note, however, if x is close to the boundaries of $\mathbb{D}$, then it might be the case that most of the volume of the sphere lies outside $\mathbb{D}$. Also, the sphere associated with x might have significant overlap with spheres of other points that are already selected. To fix this issue, we pick the point whose sphere has the largest intersection with unexplored search space in terms of its volume. The pseudo code of this method is described in Algorithm 1, which we refer to as the Next Best exploRative Sample (NBRS) algorithm. NBRS achieves the optimal exploration in the sense that it maximizes the *expected explored* volume.

The value of $|M - \mu_x| - 1.5\sigma_x$ might be negative, especially for large values of σ_x. This artifact happens at points x that are "far" from previously observed samples. To prevent/minimize this, we need to make sure that the observed samples affect the mean and variance of all points in the space. For example, if we use the Gaussian kernel $k(x_1, x_2) = \exp(-\frac{1}{\ell_r}\|x_1 - x_2\|_2^2)$ for exploration, then we need to choose ℓ_r large enough to make sure each observed sample affects all the points in the space, e.g., $\ell_r \geq \sum_{i=1}^d (b_i - a_i)^2$. If we pick small ℓ_r, then the exploration algorithm starts exploring around the previous samples and extend the explored area gradually to reach to the other side of the search space. This strategy is not optimal if we have limited samples for exploration.

To implement NBRS, we need to maximize the volume

$$g(x) = \textbf{Vol}\left(\mathbb{D}_t \cap \mathbb{S}\left(x, \frac{\left|M - \mu_{x|\mathcal{O}}\right| - 1.5\sigma_{x|\mathcal{O}}}{L}\right)\right)$$

where $\mathbb{D}_t$ represents the current unexplored input space. To evaluate $g(x)$, we take a large number of points N inside the sphere $\mathbb{S}(x, \frac{\left|M-\mu_{x|\mathcal{O}}\right|-1.5\sigma_{x|\mathcal{O}}}{L})$ uniformly at random. Then, for each point, we check if it crosses the borders $[a_i, b_i]$ or falls into the spheres of previously observed samples. If not, we count that point as a newly explored point. Finally, if there are n newly explored points, then we set $g(x) \approx \frac{n}{N}\left(\frac{\left|M-\mu_{x|\mathcal{O}}\right|-1.5\sigma_{x|\mathcal{O}}}{L}\right)^d$.

To optimize $g(x)$, one can use deterministic and derivative free optimizers like DIRECT [10]. The problem is that DIRECT only optimizes Lipschitz continuous functions; however, $g(x)$ is not necessarily Lipschitz continuous. In our implementation, we take a large number of points inside $\mathbb{D}_t$ and evaluate $g(\cdot)$ at those points and pick the maximum. This method might be slower than DIRECT, but avoids inaccurate results of DIRECT especially when $\mathbb{D}_t$ describes a small region.

3.2 Exploitation

In the exploitation phase of the algorithm, we would like to use the information gained in the exploration phase to find the optimal point of $f(\cdot)$. Suppose we have explored the search space with t samples and we want to find $x^* \in \mathbb{D}_t$. In order to exploit, we would like to find points x whose sphere is small. The reason is that if $r_x = \frac{M-f(x)}{L} \leq \gamma$ is small

Algorithm 2 Next Best exploItive Sample (NBIS)

```
Input: Maximum $M$, Lipschitz Constant $L$ and Set of observed samples $\{(x_1, f(x_1)), \ldots, (x_q, f(x_q))\}$
Output: Next best exploitive sample $x$

$\mathbb{D}_q = \mathbb{D} - \bigcup_{i=1}^{q} \mathbb{S}(x_i, r_{x_i})$
$x \longleftarrow \underset{x \in \mathbb{D}_q}{\text{argmin}}\ \textbf{Vol}\left(\mathbb{S}\left(x, \frac{|M - \mu_{x|\mathcal{O}}| + 1.5\sigma_{x|\mathcal{O}}}{L}\right)\right)$
```

enough, then by *local* strong convexity of $f(\cdot)$ around x^*, for some constant κ we have

$$\frac{\kappa}{2}\|x - x^*\|_2^2 \le M - f(x) \le L\gamma.$$

Following the argument in Section 3.1, we estimate r_x by its mean $\mathbb{E}[r_x] = \frac{|M-\mu_x|}{L}$. By Hoeffding inequality, for all $\epsilon > 0$, we have

$$\mathbb{P}\left[r_x > \frac{|M - \mu_x|}{L} + \epsilon\right] \le \exp\left(-\frac{2\epsilon^2 L^2}{\sigma_x^2}\right).$$

Similarly, replacing ϵ with $1.5\frac{\sigma_x}{L}$, the above inequality entails that with high probability (99%), $r_x \le \frac{|M-\mu_x|+1.5\sigma_x}{L}$. Hence, a "good" algorithm for exploitation should try to find the point x that minimizes the upper bound on r_x. This choice of x introduces the expected closest point to x^*. We present the pseudo code of this method in Algorithm 2.

The optimization in Algorithm 2 is nothing but minimizing

$$h(x) = \frac{|M - \mu_{x|\mathcal{O}}| + 1.5\sigma_{x|\mathcal{O}}}{L}.$$

To optimize $h(x)$, again we take a large number of points in $\mathbb{D}_q$ (the current unexplored space) uniformly at random and evaluate $h(\cdot)$ on those and pick the minimum.

3.3 Exploration-Exploitation Trade-off

The main algorithm consists of an initial exploration phase followed by exploitation. Notice that we are using GP as an estimate of the unknown function and our method, like EI, highly relies on the quality of this estimation. On a high level, if the function is very complex, i.e., has large Lipschitz constant L, then we need more exploration to fit better with GP. Small values of L correspond to flatter functions that are easier to optimize. Thus, in general, we expect the number of exploration steps to scale up with L. As a rule of thumb, functions we normally deal with satisfy $2 < L < 20$, for which we spend 20% of our budget in exploration and the rest in exploitation.

We use different kernel widths for the exploration and exploitation phases. In the case of exploration for complex functions, if we have enough budget (and hence, enough explorative samples), the kernel width can be set to a small value to fit a better local GP model. However, if we do not have enough budget, we need to take the kernel width to be large. In the case of exploitation, we pick the kernel width under which EI achieves its best performance.

Note that the choice of M and L plays a crucial role in this algorithm. If we pick L larger than the true Lipschitz function, then the radius of our spheres shrink and hence we might need more budget to achieve a certain performance. Choosing L smaller than the true Lipschitz is dangerous since it makes the spheres large and increases the chance of including the optimal point in a sphere and hence removing it. Thus, it is better to choose L slightly larger than our estimate of the true Lipschitz to be on the safe side.

The method is less sensitive to the choice of M, since the derivative of the radius with respect to M is proportional to $\frac{1}{L}$. Thus, as long as we do not over estimate M significantly, the $\frac{1}{L}$ factor prevents the spheres to become very large (and include/remove the optimal point). Small values of M, make the spheres smaller and hence, if we underestimate M, we would need more budget to achieve certain performance. However, if M is significantly (proportional to L) smaller than the true maximum of the function, then the algorithm will look for the point that achieves M and hence will perform poorly.

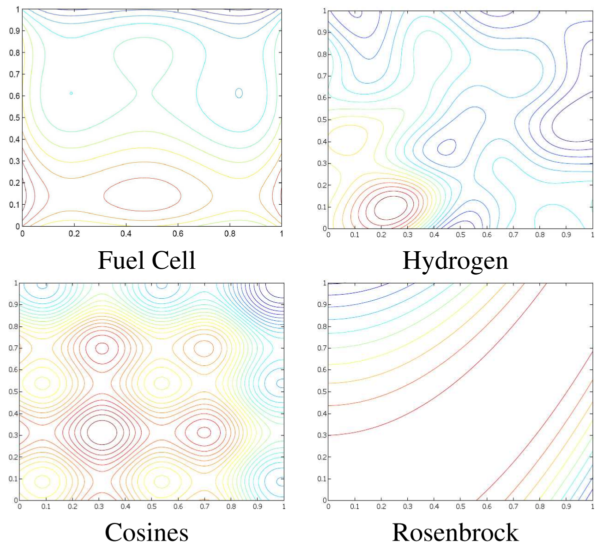

Figure 2: The contour plots for the four 2−dimension proposed benchmarks.

Table 1: Benchmark Functions

Cosines(2)	$1-(u^2+v^2-0.3\cos(3\pi u)-0.3\cos(3\pi v))$ $u=1.6x-0.5, v=1.6y-0.5$	Rosenbrock(2)	$10-100(y-x^2)^2-(1-x)^2$
Hartman(3,6)	$\sum_{i=1}^{4}\Omega_i \exp\left(-\sum_{j=1}^{d} A_{ij}(x_j-P_{ij})^2\right)$ $\Omega_{1\times 4},\ A_{4\times d},\ P_{4\times d}$ are constants	Michalewicz(5)	$-\sum_{i=1}^{5}\sin(x_i)\sin\left(\frac{i\,x_i^2}{\pi}\right)^{20}$
Shekel(4)	$\sum_{i=1}^{10}\frac{1}{\omega_i+\Sigma_{j=1}4(x_j-B_{ji})^2}$	$\omega_{1\times 10},\ B_{4a\times 10}$ are constants	

4 Experimental Results

In this section, we compare our algorithm with EI under different scenarios for different functions. We consider six well-known synthetic benchmark functions:

(1,2) Cosines [1] and Rosenbrock [5] over $[0,1]^2$

(3,4) Hartman(3,6) [7] over $[0,1]^{3,6}$

(5) Shekel [7] over $[3,6]^4$

(6) Michalewicz [14] over $[0,\pi]^5$

The mathematical expression of these functions are shown in Table 1. Moreover, we use two benchmarks derived from real-world applications:

(1) Hydrogen [6] over $[0,1]^2$

(2) Fuel Cell [3] over $[0,1]^2$

The contour plots of these two benchmarks along with the Cosines and Rosenbrock benchmarks are shown in Fig 4. The Fuel Cell benchmark is based on optimizing electricity output of microbial fuel cell by modifying some nano structure properties of the anodes. In particular, the inputs that we try to adjust are the average area and average circularity of the nano tube and the output that we try to maximize is the power output of the fuel cell. We fit a regression model on a set of observed samples to simulate the underlying function $f(\cdot)$ for evaluation. The Hydrogen benchmark is based on maximizing the Hydrogen production of a particular bacteria by varying the PH and Nitrogen levels of its growth medium. A GP is fitted to a set of observed samples to simulate the underlying function $f(\cdot)$. We consider a Lipschitz constant $L \approx 3$ for all of the benchmarks, except for Cosines and Michalewicz with $L \approx 6$ and Rosenbrock with $L \approx 45$. For the sake of comparison, we consider the normalized versions of all these functions and hence $M = 1$ in all cases. As mentioned previously, we spend 20% of the budget on exploration and 80% on exploitation.

Function	EI	NBRS+EI	NBRS+NBIS
Cosines	$.0736 \pm .016$	$.1057 \pm .029$	$\mathbf{.0270 \pm .009}$
Fuel Cell	$.1366 \pm .006$	$.1357 \pm .004$	$\mathbf{.0965 \pm .004}$
Hydrogen	$.0902 \pm .004$	$.1149 \pm .004$	$\mathbf{.0475 \pm .006}$
Rosen	$.0134 \pm .001$	$.0163 \pm .001$	$\mathbf{.0034 \pm .000}$
Hart(3)	$.0618 \pm .006$	$.0450 \pm .003$	$\mathbf{.0384 \pm 0.003}$
Shekel	$.3102 \pm .017$	$\mathbf{.3011 \pm .018}$	$.3240 \pm .030$
Michal	$.5173 \pm .010$	$.5011 \pm 0.010$	$\mathbf{.4554 \pm 0.019}$
Hart(6)	$.1212 \pm .002$	$.1235 \pm .002$	$\mathbf{.1020 \pm .003}$

Table 2: Comparison of the best results of EI, NBRS+EI and NBRS+NBIS. This result shows that our algorithm outperforms the other two counterparts significantly in most cases both in terms of the mean and variance of the performance.

4.1 Comparison to EI

In the first set of experiments, we would like to compare our algorithm with the best possible performance of EI. For each benchmark, we search over different values of the kernel width and find the one that optimizes EI's performance. Fig. 1 is plotted using these optimal kernel widths and shows that the best performance of EI happens when we take only one random sample from a given budget. This performance is then used as the baseline for comparison in Table 2. In light of the results of Fig. 1, we are also interested in whether our exploration algorithm can be used to improve the performance of EI. To this end, we replace the proposed exploitation algorithm with EI to examine if our exploration strategy helps EI. We refer to this setting as NBRS+EI.

Table 2 summarizes the mean and variance of the performance, measured as the "Regret"$= M - \max f(x_{\mathcal{O}})$, for different benchmarks estimated over 1000 random runs. It is easy to see that in all benchmarks, our algorithm (NBRS+NBIS) outperforms EI consistently except for the Shekel benchmark where EI and NBRS+EI have slightly better performances. We suspect this is due to the fact that we have not optimized our kernel widths, where as the EI kernel width is optimized.

We also note that NBRS+EI does not lead to any consistent improvement over EI. This is possibly due to the fact that EI does not take advantage of the reduced search space produced by NBRS during selection.

4.2 Exploration Analysis

In the second set of experiments, we would like to compare our exploration algorithm NBRS with random exploration when using NBIS for exploitation. As discussed previously, both random exploration and NBRS fail to produce better performance when used with EI. Thus, it is interesting to see whether they can help NBIS in terms of the overall regret, and if so which one is more effective. Figure 3 summarizes this result for all benchmarks. For a fixed budget n_b, we start with 1 explorative sample (either using NBRS or random) followed by $n_b - 1$ NBIS samples; next, we start with 2 explorative samples followed by $n_b - 2$ NBIS samples and so on. In each case, we average the regret over 1000 runs. The black line corresponds to the NBRS exploration and the green line corresponds to the random exploration. We will discuss each function in more details later, but in general, this result shows that our exploration algorithm is a) better than random exploration and b) necessary. To see why it is necessary, notice that the minimum regret on all curves is achieved for a non-zero number of NBRS samples. This means unlike EI, our exploitation algorithm benefits from NBRS.

Looking closer into the results, we see that NBRS always lead to a smaller regret comparing to the random exploration. On the Shekel benchmark, we see that random exploration has better performance if we spend majority of the budget to *explore*. However, for a *reasonable* amount of exploration that leads to the minimum regret (5 to 10 experiments), random exploration and NBRS achieve similar performance.

On our 6-dimensional benchmark Hartman(6), we notice that random exploration and NBRS behave very similarly. This shows that the input space is so large that no matter how clever you explore, you will not likely to improve the performance for the limited budget of 35.

NBRS starts from an initial point and explores the input space step by step. Imagine you are in a dark room with a torch in your hand and you want to explore the room. You start from an initial point and little by little walk through the space until you explore the whole space. This is exactly how NBRS does the exploration. Roughly speaking, NBRS minimizes $\mu_{x|\mathcal{O}} + 1.5\sigma_{x|\mathcal{O}}$ and hence, if a point is far from previous observations, i.e., $\sigma_{x|\mathcal{O}}$ is large, it is unlikely

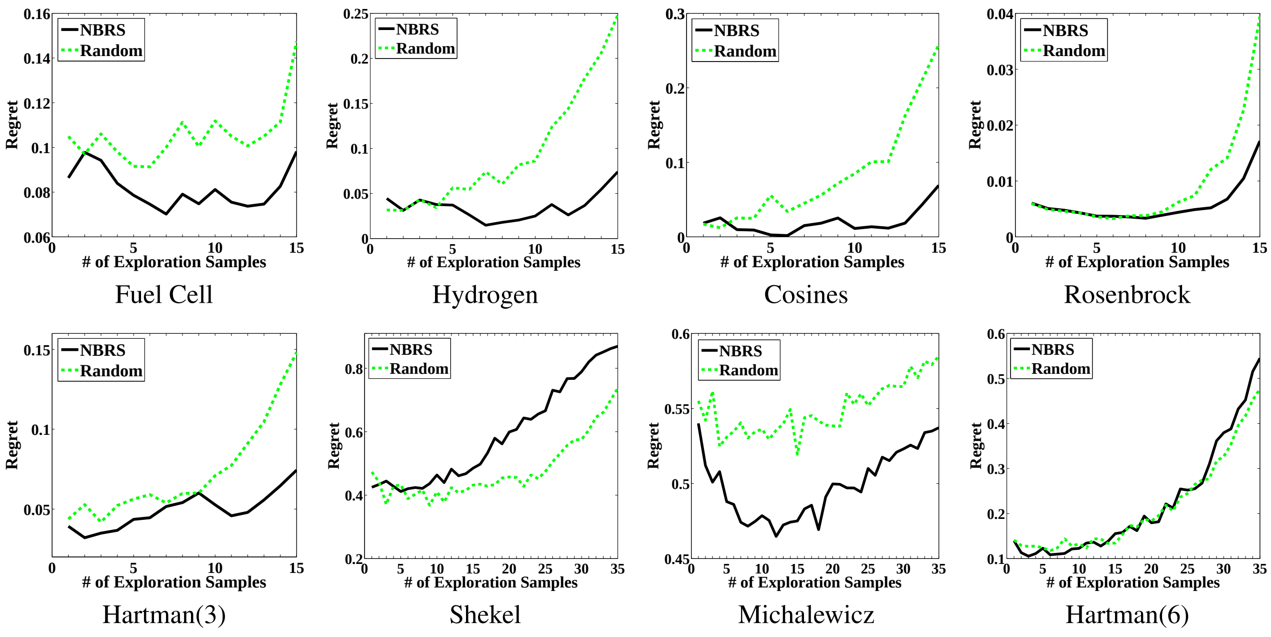

Figure 3: Plot of regret versus the number of explorations for NBIS algorithm. For a fixed budget n_b, we run a number of experiments as follows: first we consider the case where there are 1 explorative sample (either random or NBRS) followed by $n_b - 1$ EI samples, next we consider the case where where there are 2 explorative samples followed by $n_b - 2$ EI samples and so on. For 2D and 3D functions, we let $n_b = 15$ and for high-dimensional functions, we let $n_b = 35$. This result shows that in most cases, our exploration is a) better than random, and b) necessary, since the regret achieves its minimum somewhere apart from zero. On average, we need to explore 20% of our budget, however, this portion can be optimized if we consider any specific function.

to be chosen. We see this effect in all functions, but most clearly in the Michalewicz benchmark. When the number of explorative samples is smaller than 10, the step-by-step explore procedure cannot explore the whole space and the exploitation can be trapped in local minima. For $10 - 15$ explorative samples, NBRS can walk through the entire space fairly well and hence we get a minimum regret. For more than 15 explorative samples, since the space is well explored, we are wasting the samples that could be potentially used to improve our exploitation and hence, the performance becomes worse.

Finally, this investigation suggests that the result in Table 2 can be further improved by taking different number of explorative samples for different functions. To minimize parameter tuning, we chose to explore 20% of our budget. In general, this ratio can be adjusted according to the property of the function (e.g., the Lipschitz constant).

5 Conclusion

In this paper, we consider the problem of maximizing an unknown costly-to-evaluate function when we have a small evaluation budget. Using the Bayesian optimization framework, we proposed a two-phase exploration-exploitation algorithm that finds the maximizer of the function with few function evaluations by leveraging the Lipschitz property of the unknown function. In the exploration phase, our algorithm tries to remove as many points as possible from the search space and hence shrinks the search space. In the exploitation phase, the algorithm tries to find the point that is closest to the optimal. Our empirical results show that our algorithm outperforms EI (even in its best condition).

References

[1] Brigham S. Anderson, Andrew Moore, and David Cohn. A nonparametric approach to noisy and costly optimization. In *ICML*, 2000.

[2] Javad Azimi, Alan Fern, and Xiaoli Fern. Batch bayesian optimization via simulation matching. In *NIPS*, 2010.

[3] Javad Azimi, Xiaoli Fern, Alan Fern, Elizabeth Burrows, Frank Chaplen, Yanzhen Fan, Hong Liu, Jun Jaio, and Rebecca Schaller. Myopic policies for budgeted optimization with constrained experiments. In *AAAI*, 2010.

[4] Eric Brochu, Mike Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Technical Report TR-2009-23, Department of Computer Science, University of British Columbia, 2009.

[5] Mauro Brunato, Roberto Battiti, and Srinivas Pasupuleti. A memory-based rash optimizer. In *AAAI-06 Workshop on Heuristic Search, Memory Based Heuristics and Their applications*, 2006.

[6] Elizabeth H. Burrows, Weng-Keen Wong, Xiaoli Fern, Frank W.R. Chaplen, and Roger L. Ely. Optimization of ph and nitrogen for enhanced hydrogen production by synechocystis sp. pcc 6803 via statistical and machine learning methods. *Biotechnology Progress*, 25:1009–1017, 2009.

[7] L.C.W. Dixon and G.P. Szeg. *The Global Optimization Problem: An Introduction Toward Global Optimization*. North-Holland, Amsterdam, 1978.

[8] IV Elder, J.F. Global rd optimization when probes are expensive: the grope algorithm. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 577–582, 1992.

[9] D. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001.

[10] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

[11] Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001.

[12] D. Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, Edmonton, Alberta, Canada, 2008.

[13] M. Locatelli. Bayesian algorithms for one-dimensional global optimization. *Journal of Global Optimization*, 10 (1):57–76, 1997. ISSN 0925-5001. doi: http://dx.doi.org/10.1023/A:1008294716304.

[14] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs (2nd, extended ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1994. ISBN 3-540-58090-5.

[15] Andrew Moore, Jeff Schneider, Justin Boyan, and Mary Soon Lee. Q2: Memory-based active learning for optimizing noisy continuous functions. In *ICML*, pages 386–394, 1998.

[16] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT, 2006.

[17] M. J. Sasena. *Flexibility and Efficiency Enhancement for Constrained Global Design Optimization with Kriging Approximations*. PhD thesis, University of Michigan, Michigan, MI, 2002.

[18] M. Schonlau. *Computer Experiments and Global Optimization*. PhD thesis, University of Waterloo, Waterloo, Ontario, Canada, 1997.

[19] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proc. International Conference on Machine Learning (ICML)*, 2010.

[20] B. E Stuckman. A global search method for optimizing nonlinear systems. In *IEEE transactions on systems, man, and cybernetic*, volume 18, pages 965–977, 1988.

[21] E. Vazquez and J. Bect. Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *Journal of Statistical Planning and Inference*, 140(11):3088–3095, 2010.